O I P E IAP??
DEC 1 6 2005
PATENT & TRADEMARK OFFICE

H10019JDP
Customer No. 01333

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Edward Neil Chapman	Group Art Unit: 2626
PRINTING SYSTEM AND METHOD FOR CUSTOMIZATION OF A PRINT JOB	Examiner: Michael Burleson
Serial No. 09/731,503	I hereby certify that this correspondence is being deposited today with the United States Postal Service as first class mail in an envelope addressed to Commissioner For Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
Filed 06 December 2000	Debra Nowacki Debra Nowacki 12/14/05 Date

RULE 1.131 DECLARATION

I, Edward Chapman, do hereby declare that:

1. I am the inventor of the above-identified patent application.

2. Prior to May 17, 2000, I completed the invention as described and claimed in the subject application in this country, as evidenced by the following:

a. I, prior to May 17, 2000, having earlier conceived the ideas represented in Claims 1-20, attached hereto as Exhibit A, did reduce such ideas to practice as evidenced by the software code attached hereto as Exhibit B.

b. Although the software code shown in Exhibit B includes revisions, described in the software code, that were made subsequent to May 17, 2000, such software code, as it existed prior to May 17, 2000 without such revisions, was successfully executed prior to May 17, 2000 to perform one or more processes described by Claims 1, 2, 3, 4, 5, 6, 7, and 8 shown in Exhibit A.

c. The software code shown in Exhibit B, as it existed prior to May 17, 2000, was successfully executed in one or more systems according to Claims 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, and 20, shown in Exhibit A, prior to May 17, 2000.

d. The software code shown as Exhibit B, as it existed prior to May 17, 2000 and when executed as described in statements 2b and 2c, above, worked for its intended purpose.

e. I further declare under penalty of perjury pursuant to the laws of the United States of America that the foregoing is true and correct and that this declaration was executed by me on DEC 1, 2005 at Rochester, New York.

Edward N. Chapman	DEC 1, 05
Edward Neil Chapman	Date

42 Bending Creek Rd.
Street

Rochester, NY 14624
City, State, Zip

EXHIBIT A

1. A method of customizing a print job, the method comprising the steps of:

receiving an input of an application file;

selecting a preferential document-processing feature from a group of document- processing features for a print job; and

applying a plug-in module, for supporting the preferential document-processing feature, to the application file.

2. The method according to claim 1 further comprising the step of printing at least a portion of the application file using the plug-in module for the print job.

3. The method according to claim 1 wherein the application file comprises a page description language file selected from the group consisting of a portable document format (PDF), printer control language (PCL), and a PostScript file.

4. The method according to claim 1 further comprising the step of:

determining whether or not the application file represents a page description language file;

converting the received application file into a page description language file if the received application file does not represent a page description file.

5. The method according to claim 1 wherein the selecting step comprises:

accessing a plug-in module database to retrieve the selected plug-in module.

6. A method of customizing a print job, the method comprising the steps of:

receiving an input of an application file;

converting the application file into a page description language file if the application file is in a format distinct from the page description language file format;

associating a preferential document-processing feature with the page description language file;

selecting a plug-in module associated with the preferential document-processing feature for a print job; and

printing the page description language file using the selected plug-in module for a print job.

7. The method according to claim 6 wherein the page description language file is in a form selected from the group consisting of a portable document format (PDF), printer control language (PCL), and a PostScript file.

8. The method according to claim 6 wherein the selecting step comprises:

accessing a plug-in database to retrieve the selected plug-in module.

9. A system for customizing a print job, the system comprising:

a detector for receiving an input of an application file and determining whether the application file represents a page description language file;

a user interface for selecting a preferential document-processing feature

from a group of document-processing features; and

a printer for applying a plug-in module, associated with the preferential document-processing features, to the application file.

10. The system according to claim 9 where the printer includes a bitmap printing module for printing the application file.

11. The system according to claim 9 wherein the application file comprises a page description language file selected from the group consisting of a portable document format (PDF), printer control language (PCL), and a PostScript file.

12. The system according to claim 9 further comprising:

a converter for converting the application file to a page description language file if the application file does not represent a page description language file.

13. The system according to claim 9 wherein the printer includes a customization detector, a plug-in selector, and a plug-in database; the customization detector configured to detect whether customization data is associated with the application file, the plug-in selector in communication with the customization detector and the plug-in database for selecting an active plug-in module based on the customization data.

14. A system of customizing a print job, the system comprising:

a detector for receiving an input of an application file and determining whether the application file represents a page description language file;

a data augmenter for associating a preferential document-processing feature with the application file; and

a plug-in selector for selecting a plug-in module for supporting the document-processing feature.

15. The system according to claim 14 comprising:

a printer for printing the application file using the selected plug-in module.

16. The system according to claim 14 wherein the application file comprises a page description language file selected from the group consisting of a portable document file (PDF), printer control language (PCL), and a PostScript file.

17. The system according to claim 14 further comprising:

a converter for converting the application file to a page description language file if the application file does not represent a page description language file.

18. The system according to claim 14 wherein the plug-in selector is adapted to access a plug-in database to retrieve the selected plug-in module.

19. The system according to claim 14 wherein the data augmenter cooperates with a downloader to express the preferential document-processing feature as downloader-embedded customization data in the application file.

20. The system according to claim 14 wherein the data augmenter cooperates with a printer driver to express the preferential document-processing feature as printer-driver-embedded customization data in the application file.

EXHIBIT B

```
/***********************************************************************
**
*
*            Copyright Heidelberg Digital L.L.C. 1999-2002
*                         ALL RIGHTS RESERVED
*
***********************************************************************
*/

/***********************************************************************
**
*
*   FILE NAME:      CheckCustom.c
*
*   SCCS Release:   @(#)CheckCustom.c     1.19     %w
*   Newest Delta:   [redacted]
*
*   FUNCTION LIST:  CheckCustom()
*                   DetermineDLs()
*                   CheckHost() -- ifdef'ed out
*
*   GENERAL DESCRIPTION:    Determine if the customlib should be called
*                           by in order check doc requirements which
*                           came from a %KDKCustom in the job, command
*                           line or environmental variables, and the
*                           default from the config utility. return TRUE
*                           if it should be called. Fill in the proper
*                           list of string and function arguments.
*                           the calling function responsibility to alloc
*                           space for the StringToCall pointer.
*
*   REVISION HISTORY:
*
*   DATE      AUTHOR            FUNCTIONS/DATA MODIFIED
*   ====      ======            ======================
*    [redacted]                 CheckCustom to own lib, 6.x interfaces
*                               strtok to strtok_r for MT safe
*                               pulled out [redacted]
*                               updated actons MAX_CUSTOM_FUNC was string
*                               5.x interfaces
*                               ifdef CheckHost since [redacted]
*                                           fixed bug in [redacted]
[redacted]
*                               Fix G_CANCEL_JOB message.
*                            Added CancelJob().
*                            Added parser for DL's and args.
*                            strcpy to strncpy
*                            moved [redacted] features to CheckHost()
*                               this version has some differences from the
[redacted]
*                            print out "NULL" for custom arg + Ts_Doc_Complete
*                            original
*
*   ADD HISTORY TO TOP
*
***********************************************************************
*/
```

```
/** INCLUDE FILES **/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>
#include <ctype.h>
#include "common.h"
#include "tslib.h"
[redacted]
#include "ts_custom.h"

/** DEFINES **/

#ifdef RIPVERSION2
#define RIPVERSION '2'
#endif
#ifdef RIPVERSION3
#define RIPVERSION '3'
#endif
#ifdef RIPVERSION4
#define RIPVERSION '4'
#endif
#ifdef RIPVERSION5
#define RIPVERSION '5'
#endif
#ifdef RIPVERSION6
#define RIPVERSION '6'
#endif

/** TYPEDEFS **/

/** MACRO DEFINITIONS **/

/** EXTERN FUNCTION DECLARATIONS **/

/** EXTERN DATA DECLARATIONS **/

/** GLOBAL FUNCTION DECLARATIONS **/

/** GLOBAL DATA DEFINITIONS **/
static T_Bool JobCanceled;
static void *dl_handles[MAX_CUSTOM_LIBS]; /* ptr's for dynamic library handle
*/

/** LOCAL FUNCTION DECLARATIONS **/
[redacted]
    char        *StringArg,
    char        **FunctionsToUse,
    char        **StringsToUse,
    void        (**StartProcsToCall)(Doc_Requirements *, char *),
    int         (**ImageProcsToCall)(Ts_Image_Complete *, char *),
    void        (**EndProcsToCall)(    Doc_Requirements *,
                                        Ts_Doc_Complete *,
                                        char *) );

#ifdef SECURITY
static int CheckHost( void );
#endif

static void CancelJob( void );
```

[redacted]

```
/** LOCAL DATA DEFINITIONS **/
```

```
/***************************************************************************
**
*
*   FUNCTION NAME:      CheckCustom
*
*   RETURN VALUE:       none
*
*   FORMAL ARGUMENTS
*                       pdr = document requirements
*                       CommandLine = TRUE if set via command line/enviroment
*                       CommandLineString = string if CommandLine TRUE
*                       FunctionsToUse = tells calling process function(s)
name
*                                       (NULL terminated list)
*                       StringsToUse = tells calling process string(s) to use
*                       StartProcsToCall = procedures to call at TS job start
*                       ImageProcsToCall = procedures to call for each image
*                       [redacted]
*
*   IMPLICIT INPUTS/OUTPUTS:
*
*   DESCRIPTION:
*
*   REVISION HISTORY:
*
*   DATE        AUTHOR      DESCRIPTION OF CHANGE
*   ====        ======      ======================
*   [redacted]               original
*
***************************************************************************
*/
T_Bool
CheckCustom  (
[redacted]
    T_Bool                   CommandLine,
    char                    *CommandLineString,
    char                    **FunctionsToUse,
    char                    **StringsToUse,
    void                    (**StartProcsToCall)(Doc_Requirements  *, char *),
    int                     (**ImageProcsToCall)(Ts_Image_Complete *, char *),
    void                    (**EndProcsToCall)  (Doc_Requirements  *,
                                                    Ts_Doc_Complete *,
                                                    char *) )
{
    static char __PROC__[] = "CheckCustom()";
    T_Bool      CallCustom;

    ts_printf(TS_DB_PROC)("%s %s - entry\n", __HEAD__, __PROC__);

    ts_printf(TS_DB_INFO)
        ("%s %s - KDK Act ON for Custom Lib = %s\n",
        __HEAD__, __PROC__,
            print_acton(pdr->kdk_acton_req.ka_Custom_lib));

[redacted]
        ("%s %s - Doc Requirements Use Custom Lib = %s\n",
        __HEAD__, __PROC__,
            /*CONSTCOND*/
        pdr->use_custom_lib == FALSE ? "FALSE" :
```

```
        pdr->use_custom_lib == TRUE ? "TRUE" :
        "UNKNOWN");

    if(pdr->custom_lib_arg == (char *)NULL)
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Doc Requirements Custom string arg = NULL\n",
            __HEAD__, __PROC__);
    }
    else
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Doc Requirements Custom string arg = %s\n",
            __HEAD__, __PROC__,
            pdr->custom_lib_arg);
    }

    ts_printf(TS_DB_INFO)
        ("%s %s - Command Line argument(-c) is %s\n",
        __HEAD__, __PROC__,
      /*CONSTCOND*/
        CommandLine == FALSE ? "FALSE" :
        CommandLine == TRUE ? "TRUE" :
        "UNKNOWN");

[redacted]
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Command Line string argument = NULL\n",
[redacted]
        ;
    }
    else
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Command Line string argument = %s\n",
            __HEAD__, __PROC__, CommandLineString);
        ;
    }

    /*  If ACT ON is set with a PL_TEMP, PL_USER_MODIFY PL_HEADER PL_PDL or
            PL_RESTORE_VALUE we will do what ever is requested by the
        doc requirements use_custom_lib field - either call the Custom lib
        or not. If Act ON is false we then check the -c command line option
        and if that is set we call the Custom lib. Finally, if both cases
        preceding are false we check the doc requirements use_custom_lib
        field. This reflects the system default.
    */

    if(pdr->kdk_acton_req.ka_Custom_lib)
    {
            /*CONSTCOND*/
        if(pdr->use_custom_lib == TRUE)
        {
            ts_printf(TS_DB_INFO)
                ("%s %s - Calling Custom() due to %KDKCustom:\n",
            __HEAD__, __PROC__);

#ifdef SECURITY
[redacted]
            {
```

```
                        return(0);
                }
#endif

                        /*CONSTCOND*/
[redacted]
                DetermineDLs(   pdr->custom_lib_arg,
                                FunctionsToUse,
                                StringsToUse,
                                StartProcsToCall,
                                ImageProcsToCall,
                                EndProcsToCall);
        }

        else
        {
                ts_printf(TS_DB_INFO)
                    ("%s %s - *NOT* Calling Custom() due to %%KDKCustom:\n",
                    __HEAD__, __PROC__);
                ;
                CallCustom = FALSE;
        }

    }

      /*CONSTCOND*/
    else if (CommandLine == TRUE)
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Calling Custom() due to command line arg\n",
            __HEAD__, __PROC__);

#ifdef SECURITY
[redacted]
        {
                return(0);
        }
#endif

                /*CONSTCOND*/
[redacted]
        DetermineDLs(   CommandLineString,
                        FunctionsToUse,
                        StringsToUse,
                        StartProcsToCall,
                        ImageProcsToCall,
                        EndProcsToCall);
    }

        /*  if act on is off but this is true it was set via the
            config utility  */
      /*CONSTCOND*/
    else if(pdr->use_custom_lib == TRUE)
    {
        ts_printf(TS_DB_INFO)
            ("%s %s - Calling Custom() due to sys default:\n",
            __HEAD__, __PROC__);

#ifdef SECURITY
        if(CheckHost() == 0)
        {
```

```
                    return(0);
        }
#endif

            /*CONSTCOND*/
        CallCustom = TRUE;

[redacted]
                                FunctionsToUse,
                                StringsToUse,
                                StartProcsToCall,
                                ImageProcsToCall,
                                EndProcsToCall);
    }

    else
    {
[redacted]
        ("%s %s - *NOT* Calling Custom()\n", __HEAD__, __PROC__);
        CallCustom = FALSE;
    }

    ts_printf(TS_DB_PROC)("%s %s - exit\n", __HEAD__, __PROC__);

    return(CallCustom);
}
```

```
/***********************************************************************
**
*
*   FUNCTION NAME:      DetermineDLs
*
*   RETURN VALUE:       none
*
*   FORMAL ARGUMENTS    StringArg = string to parse
*                       FunctionsToUse = tells calling process function(s)
name
*                                       of the ImageProcsToCall. SInce it is
*                                       a NULL terminated list it also lets
you
*                                       know how many calls to make
*                       StringsToUse = tells calling process string(s) to use
*                                       can be NULL in middle of list
[redacted]
*                                       can't be NULL in middle of list
*                       ImageProcsToCall = function(s) to call for each image
*                                       can't be NULL in middle of list
*                       EndProcsToCall = function(s) to call at job end
*                                       can't bé NULL in middle of list
*
*   IMPLICIT INPUTS/OUTPUTS:
*
*   DESCRIPTION:        Extract DLs from KDKCustom: string if present.
*
*   REVISION HISTORY:
*
*   DATE        AUTHOR      DESCRIPTION OF CHANGE
*   ====        ======      ======================
*   [redacted]               Added dl-parser.
*                            original
*
************************************************************************
*/
static void
DetermineDLs(
                char     *StringArg,
                char    **FunctionsToUse,
                char    **StringsToUse,
                void      (**StartProcsToCall)(Doc_Requirements *, char *),
                int       (**ImageProcsToCall)(Ts_Image_Complete *, char *),
                void      (**EndProcsToCall)  (Doc_Requirements *,
                                                Ts_Doc_Complete  *,
                                                char *)
            )
{

    int i = 0;                      /* generic counter */
    char *startptr; /* start point for parsing of string */

[redacted]
    char *subtokptr;

    char *tmpptr;
    char *tmpptr2;

    char *dl_args[MAX_CUSTOM_LIBS]; /* DL'-string-args array */
char  buf[MAX_CUSTOM_STRING];

T_Bool USE_DLL = FALSE;

char bc[128]; /* back channel error strings */

static char __PROC__[] = "DetermineDLs()";

JobCanceled = FALSE;

ts_printf(TS_DB_PROC)("%s %s - entry\n", __HEAD__, __PROC__);

    /*  if no -d function is Custom | CustomEndOfJob | CustomStartOfJob
        It is Custom vs. CustomImage for backwards compatibility
        if format is -d e.g. "-dStore:"arg 1";Mail;CleanUp:arg2"
        first function is KDKCustomStoreStartOfJob |
        KDKCustomStoreEndOfJob | KDKCustomStoreImage and the
        first string is "arg 1" - the 2nd function is
        KDKCustomMailStartOfJob etc with a NULL string and 3rd
        KDKCustomCleanUpStartOfJob with "arg2" as the string.
        An alias file will allow the site to remap oprions
        such as alias s store or alias s store:helloworld
    */
strncpy(StringsToUse[0], StringArg, MAX_CUSTOM_STRING);

if(StringsToUse[0] == (char *)NULL)
{
    ts_printf(TS_DB_INFO)
[redacted]
}
else
{
    ts_printf(TS_DB_INFO)
        ("%s\tFunction-Arg string = (%s)\n", __HEAD__, StringsToUse[0]);
}

i = 0;
while ( i < MAX_CUSTOM_LIBS )  /* Init Procs */
{
[redacted]
    ImageProcsToCall[i] = DEFAULT_IMAGE;
    EndProcsToCall[i]   = DEFAULT_ENDOFJOB;
    i++;
}

/* init 1st for DEFAULT use of custom */
strncpy(FunctionsToUse[0], STRING_IMAGE, MAX_CUSTOM_FUNC);

if ( (startptr = strstr(StringArg, "-d")) != (char *) NULL )
  {
        /*CONSTCOND*/
    USE_DLL = TRUE;
    ts_printf(TS_DB_INFO)("%s %s - -d Dynamic mode found! Using DLs.\n",
                __HEAD__,__PROC__);
}
else
{
    i = 1;
    while ( i < MAX_CUSTOM_LIBS ) /* skip the remaining functions */
    {
        *FunctionsToUse[i] = '\0';
```

```
            i++;
        }
    }

[redacted]
        ("%s\tDEFAULT: Using Function = (%s)\n", __HEAD__,
FunctionsToUse[0]);

    ts_printf(TS_DB_INFO)("%s\tString-Arg = (%s)\n",
          __HEAD__, StringArg);

    /*
     * Parse KDKcustom: (string)
     */

    /* Begin if dynamic lib mode */
[redacted]
    {
            char *lasts;

        startptr += strlen("-d"); /* bump up past '-d' */

            /************************************
             *      Parse KDKCustome string      *
             ************************************/

        /*
         * Break up input string by: ';'
         * "dl1:"args1 args1b";dl2:args2;..."
         * strings delim by ';'
         */
        i = 0;
        while( ((tokptr = strtok_r(startptr, ";", &lasts )) != NULL) && i <
MAX_CUSTOM_LIBS ) /* clip off -d */
        {
            dl_args[i] = tokptr;

            ts_printf(TS_DB_INFO)("%s\tdl-args[%d] = (%s)\n",
                    __HEAD__, i, tokptr);

[redacted]
            i++;

        }


        while ( i < MAX_CUSTOM_LIBS ) /* NULL out the rest of dl_args array
if no more functions */
        {
[redacted]
            i++;
        }

        /*
         *  Break up "dl:args" string by ":" to get "dl-args".
         */

        i = 0;
        while( ((tokptr = strtok(dl_args[i], ":")) != NULL)
                && i < MAX_CUSTOM_LIBS
                && dl_args[i] != (char *) NULL)
        {
                tmpptr  = FunctionsToUse[i];
                tmpptr2 = tokptr;

            /*
             * Strip off white-space
             */
            while( !(*tmpptr2 == '\0') )
            {
                if( !isspace(*tmpptr2) )
                *(FunctionsToUse[i]++) = *tmpptr2;
                tmpptr2++;
            }
            *(FunctionsToUse[i]) = '\0';
[redacted]


            ts_printf(TS_DB_INFO)("%s\tFunctionsToUse[%d] = (%s)\n",
                    __HEAD__, i, FunctionsToUse[i]);


            subtokptr = strtok((char *)NULL, ":"); /* get other half of
string */


[redacted]
                __HEAD__, i, subtokptr?subtokptr:"NULL");

            if ( subtokptr != (char *) NULL )
            {

                if ( strlen(subtokptr) == 0)
                {
                    *StringsToUse[i] = '\0';
                }
                else
                {
                    strncpy( StringsToUse[i], subtokptr, MAX_CUSTOM_STRING );
                }
            }
            else
                *StringsToUse[i] = '\0';

            i++;
        }


        while ( i < MAX_CUSTOM_LIBS ) /* NULL out the rest of array if no
more functions */
        {
[redacted]
            i++;
        }

        /*
         *  Check and open dl's.
         */
        i = 0;
        while ( (FunctionsToUse[i] != (char *)NULL) &&
                (strlen(FunctionsToUse[i])  != 0)   &&
[redacted]
```

```
        {
            sprintf(buf, "/hp/lib/KDKCustom%s.so", FunctionsToUse[i] );

            if ((dl_handles[i] = dlopen(buf, RTLD_LAZY)) == NULL)
            {

                sprintf(bc, "dlopen: %s\n", dlerror() );

                ts_printf(TS_DB_ERRORS)("%s %s - %s\n",
                    __HEAD__, __PROC__, bc);

                LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

                *FunctionsToUse[i] = '\0';

                CancelJob();

                break;

            }
            else
                ts_printf(TS_DB_INFO)
                        ("%s\tFound DLL(%s)\n", __HEAD__, FunctionsToUse[i]);


        /*
[redacted]
         * and assign them to Procs.
         *
         * Note: Use program "cdecl" to read cast
         *
         */

            if ( (StartProcsToCall[i] =
[redacted]
"CustomStartOfJob")) == NULL )
            {
                sprintf(bc, "dlsym: %s\n", dlerror());

                ts_printf(TS_DB_ERRORS)("%s %s -\n\t%s\n",
                    __HEAD__, __PROC__, bc );


                LogBackChannel(SaveQueue((long)0), bc, strlen(bc));

                StartProcsToCall[i] = '\0';

                CancelJob();

                break;
            }
            else
                ts_printf(TS_DB_INFO)("%s\tFound (%s):CustomStartOfJob()\n",
                        __HEAD__, FunctionsToUse[i]);


            if ( (ImageProcsToCall[i] =
                (int (*)(Ts_Image_Complete *, char *)) dlsym(dl_handles[i],
                        "Custom")) == NULL )
            {
[redacted]
```

```
                    ts_printf(TS_DB_ERRORS)("%s %s -\n\t%s\n",
                        __HEAD__, __PROC__, bc );

                    LogBackChannel(SaveQueue((long)0), bc, strlen(bc));

                    ImageProcsToCall[i] = '\0';

[redacted]

                    break;

                }
                else
                    ts_printf(TS_DB_INFO)
                            ("%s\tFound (%s):Custom()\n", __HEAD__,
FunctionsToUse[i]);


                if ( (EndProcsToCall[i] =
                    (void (*)(Doc_Requirements *,
                            Ts_Doc_Complete *, char *)) dlsym(dl_handles[i],
                            "CustomEndOfJob")) == NULL )
                {
                    sprintf(bc, "dlsym: %s\n", dlerror());

                    ts_printf(TS_DB_ERRORS)("%s %s -\n\t%s\n",
                        __HEAD__, __PROC__, bc );

                    LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

                    EndProcsToCall[i] = '\0';

                    CancelJob();

[redacted]

                }
                else
                    ts_printf(TS_DB_INFO)("%s\tFound (%s):CustomEndOfJob()\n",
__HEAD__, FunctionsToUse[i]);

                i++;

            } /* End of while FunctionsToUse[i] */

        } /* End if dynamic lib mode */

[redacted]
}
```

```
/*****************************************************************************
**
*
*   FUNCTION NAME:      CheckHost
*
*   RETURN VALUE:       none
*
*   FORMAL ARGUMENTS
*
*   IMPLICIT INPUTS/OUTPUTS:
*
*   DESCRIPTION:
*
*   REVISION HISTORY:
*
*   DATE        AUTHOR      DESCRIPTION OF CHANGE
*   ====        ======      ======================
*   [redacted]  original
*
*****************************************************************************
*/
#ifdef SECURITY
static int
CheckHost( void )
{
    static char __PROC__[] = "CheckHost()";
    static char     command[] = "/usr/bin/pkginfo -l KDKrip | /usr/bin/grep
VERSION | awk '{ print $2 }'";
    unsigned long   hostid;
    unsigned long   validhostid;
#if 1 /* swich back if if 0 */
    static char     hoststr[] = "0X00EDEDED";
#else
      static char hoststr[] = "0X80a6b329";     /*    EdC         */
[redacted]
      static char hoststr[] = "0X808cf598";     /*    dhvem21     */
#endif

    FILE            *the_pipe;
    unsigned char   buffer[ARG_SIZE];

    ts_printf(TS_DB_PROC)("%s %s - entry\n", __HEAD__, __PROC__);

    hostid = (unsigned long)gethostid();

    ts_printf(TS_DB_INFO) ("%s %s -     validating hostid = 0X%x\n",
        __HEAD__, __PROC__, hostid);

[redacted]

    if (hostid != validhostid)
    {
        char    bc[128];

        sprintf (bc, "%s Invalid hostid expected: 0x%x\n", VERSION,
validhostid);

        ts_printf(TS_DB_INFO) ("%s %s - %s\n", __HEAD__, __PROC__, bc);

        LogBackChannel (SaveQueue((long)0), bc, strlen(bc));
```

```
            /*  this will be unknown error until 03.00   */
        ts_interp_error(     (int)SaveQueue((long)0),
                             TS_WARN_HOST_ID,
                             __PROC__,
                             (char *)NULL,
                             FALSE);
```

[redacted]

```
        return(0);
    }

        /*   now check if RIP version number is ok   */

    ts_printf(TS_DB_INFO) ("%s %s -     validating RIP version\n",
        __HEAD__, __PROC__);

    memset(buffer, '\0', ARG_SIZE);

    the_pipe = popen(command,"r");

    if(fread(buffer, 1, ARG_SIZE, the_pipe) == 0)
    {
        char    bc[128];
```

[redacted]

```
        ts_printf(TS_DB_INFO) ("%s %s - %s\n", __HEAD__, __PROC__, bc);

        LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

            /*  this will be unknown error until 03.00   */
        ts_interp_error(     (int)SaveQueue((long)0),
                             TS_WARN_POPEN,
                             __PROC__,
                             (char *)NULL,
                             FALSE);

        ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);
        return(0);
    }
```

[redacted]

```
    ts_printf(TS_DB_INFO) ("%s %s - RIP version = %s\n",
        __HEAD__, __PROC__, buffer);

        /* this means works with any 03.xx.xx.xxxx version   */
    if((buffer[0] != '0') || (buffer[1] != RIPVERSION))
    {
        char    bc[128];

        sprintf (bc, "%s Invalid RIP version\n", VERSION);

        ts_printf(TS_DB_INFO) ("%s %s - %s\n", __HEAD__, __PROC__, bc);

        LogBackChannel (SaveQueue((long)0), bc, strlen(bc));

            /*  this will be unknown error until 03.00   */
            ts_interp_error( (int)SaveQueue((long)0),
```

[redacted]

```
                                        __PROC__,
                                        (char *)NULL,
                                        FALSE );

        ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);
        return(0);
    }

    return(1);
}
#endif      /* SECURITY */
```

```
/***********************************************************************
**
*
*   FUNCTION NAME:      CancelJob()
*
*   RETURN VALUE:       none
*
*   FORMAL ARGUMENTS
*
*   IMPLICIT INPUTS/OUTPUTS:
*
*   DESCRIPTION: Will cancel current job.
*
***********************************************************************
*/
static void
CancelJob(void)
{
    Uint32 jobid;

[redacted]

    ts_printf(TS_DB_PROC) ("%s %s - entry\n", __HEAD__, __PROC__);

    if(JobCanceled)
    {
        ts_printf(TS_DB_PROC) ("%s %s - (job previously canceled) exit\n",
__HEAD__, __PROC__);
        return;
    }
      else
            /*CONSTCOND*/
        JobCanceled = TRUE;


    jobid = GetJobID();


    pgcj = (G_Cancel_Job *)valloc(sizeof(G_Cancel_Job));
[redacted]
    {
        ts_printf(TS_DB_PROC) ("%s %s - error malloc()\n", __HEAD__,
__PROC__);

        ts_interp_error( (int)SaveQueue((long)0),
                                    TS_ERR_MALLOC,
                                    __PROC__,
                                    (char *)NULL,
                                                  /*CONSTCOND*/
                                    TRUE );
    }


    pgcj->ack         = (int)0;
    pgcj->response    = (int)0;
    pgcj->operation   = (int)0;
    pgcj->job_id      = (Uint32)jobid; /* Cancel last job */
    pgcj->connect_id  = (long)0;
    ts_printf(TS_DB_PROC) ("%s %s - Canceling Job-%d\n",
                __HEAD__, __PROC__, jobid);
```

[redacted]

```
    free( (G_Cancel_Job *)pgcj );
    pgcj = (G_Cancel_Job *)NULL;

    ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);

    return;
}
```

```
/***************************************************************************
**
*
*    FUNCTION NAME:        CloseCustom()
*
*    RETURN VALUE:         none
*
*    FORMAL ARGUMENTS
*
*    IMPLICIT INPUTS/OUTPUTS:
*
*    DESCRIPTION: Close Custom dll handles.
*
****************************************************************************
*/
void
CloseCustom(void)
{
    char bc[128]; /* back channel error strings */
    int i;
    static char __PROC__[] = "CloseCustom()";

    ts_printf(TS_DB_PROC) ("%s %s - entry\n", __HEAD__, __PROC__);

    i = 0;
    while( (dl_handles[i] != (void *)NULL) &&  i < MAX_CUSTOM_LIBS )
    {
        ts_printf(TS_DB_ERRORS)("%s\t Closing handle %d\n",
                __HEAD__,i);

        if( dlclose(dl_handles[i]) != 0 )
        {
[redacted]
            ts_printf(TS_DB_ERRORS)("%s %s - %s\n",
                __HEAD__, __PROC__, bc);

            LogBackChannel(SaveQueue((long)0), bc, strlen(bc));

            CancelJob();

        }
[redacted]
        i++;
    }

    ts_printf(TS_DB_PROC) ("%s %s - exit\n", __HEAD__, __PROC__);

    return;
}
```